МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет имени
первого Президента России Б. Н. Ельцина»

ВЫДЕЛЕНИЕ ТРЕНДОВОЙ СОСТАВЛЯЮЩЕЙ ВРЕМЕННОГО РЯДА

Методические указания к выполнению практического задания № 3

Екатеринбург
2024

Содержание

Введение

На этой лабораторной работе мы впервые приступаем к декомпозиции временных рядов на простейшие компоненты, одной из которых является **тренд**. В ходе работы студентами будут изучены такие способы построения кривых тренда, как регрессионные методы подгонки, методы скользящего сглаживания, и другие.

1. Задание на лабораторную работу

Результатом выполнения лабораторной работы является оформленный отчет в виде *Jupyter*-тетради, в котором должны быть представлены и отражены все нижеперечисленные пункты:

1) Сначала импортируйте в свой код нужные библиотеки, функции и т.д.

```
import numpy as np
import numpy.random as rand
import matplotlib.pyplot as plt
import pandas as pd
from scipy import signal
import scipy.stats as stats
from statsmodels.tsa import api as tsa
```

2) В зависимости от своего варианта, который определяется по последним двум цифрам студ. билета, создать временной ряд из таблицы. ВР определен на временном интервале от 0 до 1 (далее переменная t). Для загрузки своего варианта используйте:

```
table = pd.read_excel('for_lab3.xlsx')
variant = 13                # номер варианта, например, 13
Y = np.array(table.values[variant-1][1:])
print(Y)
```

3) Построить график заданного ряда.
4) Рассчитать регрессионную модель тренда первого порядка, то есть линейный тренд $\tau(t) = \beta_0 + \beta_1 t$. Для этого:
5) Сначала произвести оценку регрессионной модели $y = X\beta$.

   Для этого потребуется в матричном виде решить эту систему линейных уравнений.

6) Для линейного тренда $X = \begin{vmatrix} 1 & t_1 \\ 1 & t_2 \\ \vdots & \vdots \\ 1 & t_N \end{vmatrix}, y = \begin{vmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{vmatrix}, \beta = \begin{vmatrix} \beta_0 \\ \beta_1 \end{vmatrix}$.
7) Для решения Вам пригодится функция:

   B = np.linalg.lstsq(X,Y)

8) Из этого результата **B** коэффициенты находятся в нулевом элементе. Построим получившийся тренд:

```
B = B[0]    # забираем из результата коэффициенты β
print(B)
plt.figure(figsize = (10, 5))
plt.plot(t, Y)                    # строим исходный ВР
plt.plot(t, B[0] + B[1] * t , 'r')    # строим его тренд
plt.show()
```

9) Кроме матричных расчетов в Python, несомненно, существуют и готовые функции построения регрессионных кривых. Воспользуемся ими из нескольких библиотек.

10) На основе построения полиномиальных кривых из **numpy**:

```
bb = np.polyfit(t, Y, 1) # полиномиальная кривая 1-го порядка
plt.figure(figsize = (10, 5))
plt.plot(t, Y)
plt.plot(t, bb[1] + bb[0]*t, 'r') # Внимание! Коэф. β в другом порядке
plt.show()
```

Чтобы не ошибиться в порядке коэффициентов, лучше использовать функцию **poly1d**:

```
p = np.poly1d(bb)       # создаем экземпляр полинома
plt.figure(figsize = (10, 5))
plt.plot(t, Y)
# считаем значения полинома на заданной временной сетке
plt.plot(t, p(t), 'g')
plt.show()
```

11) На основе линейной регрессии из **scipy.stats**:

```
out = stats.linregress(t, Y)
print(out)          # выведет все коэффициенты и статистику регрессии
plt.figure(figsize = (10, 5))
plt.plot(t, Y)              # строим график кривой вместе с трендом
plt.plot(t, out.intercept + out.slope*t, 'r')
plt.show()
```

12) На основе подгонки кривых **curve_fit** из **scipy.optimize**:

```
def func(t, b0, b1):      # описываем функцию тренда
    return b0 + b1 * t    # линейный тренд с 2 параметрами

from scipy.optimize import curve_fit
popt, pcov = curve_fit(func, t, Y)    # проводим подгонку МНК
print(popt)       # получаем коэффициенты b0 & b1
print(pcov)       # ковариационная матрица ошибок подгонки
```

13) На основе библиотеки **sklearn**:

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(t.reshape(-1,1), Y)
print(reg.coef_)          # здесь выведется линейный коэффициент b1
print(reg.intercept_)     # здесь выведется коэффициент b0 (смещение)
print(reg.score(t.reshape(-1,1), Y))
# здесь будет выведена «оценка» (равная R^2) полученной регрессии,
чем ближе она к 1.0, тем лучше тренд
```

14) На основе **statsmodel**:

```
import statsmodels.api as sm
x_ = sm.add_constant(t.reshape(-1,1)) # создаем простую модель
                                      # вида τ(t) = β0 + β1t
smm = sm.OLS(Y, x_)   # используем Метод Наименьших Квадратов
                      # (МНК) (Ordinary Least Squares = OLS)
res = smm.fit()       # подгоняем параметры модели по МНК
print(res.params)     # получаем результирующие коэффициенты
```

15) Пример, приведенный выше, гораздо ближе по своей реализации уже к методам **машинного обучения**: сначала задается «форма» решаемой задачи, затем определяется метод ее решения и уже самим решением занимается ЭВМ.

Удостоверьтесь, что во всех реализациях получились одинаковые коэффициенты линейного тренда. Постройте для каждого из методов графики ВР и линейных трендов.

16) Аналогичным образом постройте модель тренда **второй**, **третьей, четвертой, пятой, шестой, седьмой и восьмой** степени. Не забывайте про рисунки.

17) Теперь постройте модель **экспоненциального** тренда (2 версии):

$\tau(t) = \beta_0 e^{\beta_1 t}$ и $\tau(t) = \beta_0 e^{\beta_1 t} + \beta_2$

с помощью метода на основе функции **curve_fit**.

18) Постройте модель **экспоненциального** тренда $\tau(t) = \beta_0 e^{\beta_1 t}$, но только с помощью матричного решения по методу наименьших квадратов (т.е. используя функцию **B = np.linalg.lstsq(X,Y)**). Полученное решение не будет совпадать с решением из п. 17 выше. Почему? Поясните.

19) Постройте модель **логарифмического** тренда: $\tau(t) = \beta_0 \log(\beta_1 t) + \beta_2$ любым из доступных способов.

20) В конце создайте рисунок, где будут представлены все найденные тренды разной степени на одном/двум графиках.

21) Теперь построим тренд методом сглаживания. Для этого напишите следующую функцию:

```
def smooth(x, window_len):
    if window_len<3:
        return x
    s=np.r_[2*x[0]-x[window_len-1::-1], x, 2*x[-1]-x[-1:-window_len:-1]]
    w=np.ones(window_len, 'd')
    y=np.convolve(w/w.sum(), s, mode='same')
    return y[window_len:-window_len+1]
```

22) Затем вызовите эту функцию для сглаживания ряда, например:

```
Smoothed_data = smooth(Y, 3) # сглаживание по 3 точкам
```

23) Постройте тренды, полученные методом скользящего сглаживания по **трем, семи** и **одиннадцати** точкам. Постройте каждый из них **отдельно**, но вместе с исходным ВР (то есть всего 3 рисунка по 2 графика в каждом).

24) Постройте **собственную функцию** сглаживания по **семи** точкам, на основе формул из лекции 4. Сравните получившиеся результаты.

25) Наконец, постройте тренд методом **экспоненциального сглаживания**, самостоятельно подобрав его параметр (который лежит в диапазоне от 0 до 1).

26) Не забудьте в отчет-тетрадь добавить необходимые рисунки.

2. Требования к оформлению отчета

Отчет в Jupyter-тетради должен обязательно содержать: номер лабораторной работы, ФИО студента, номер варианта (либо студенческий номер), номер группы, результаты выполнения работы с комментариями студента (комментарии пишутся после #) и изображениями.